# Significance-Driven Data Truncation for Preventing Timing Failures

Ioannis Tsiokanos, Lev Mukhanov, Dimitrios S. Nikolopoulos and Georgios Karakonstantis
Institute of Electronics, Communications and Information Technology (ECIT)
School of EEECS, Queen's University Belfast
Email: {itsiokanos01, l.mukhanov, d.nikolopoulos, g.karakonstantis}@qub.ac.uk

***Abstract*—The continuous scaling of transistor sizes and the increased parametric variations render nanometer circuits more prone to timing failures. To protect circuits from such failures, typically designers adopt pessimistic timing margins, which are estimated under rare worst-case conditions. In this paper, we present a technique that mitigates such pessimistic margins by minimizing the number of timing failures. In particular, we propose a method that minimizes the number of long latency paths within each processor pipeline stage and constraints them in as few stages as possible. Such a method allows us not only to reduce the timing failures, but also to limit the potential error-prone locations to only few pipeline stages. To further reduce these failures, we exploit the path excitation dependence on data patterns and truncate the bitwidth of the operands in the few remaining long latency paths by setting a number of less significant bits (LSBs) to a constant value of zero. Such a truncation may incur quality loss, but this is limited since it is applied only to the LSBs of the few operands that may activate the confined error-prone long latency paths. To evaluate the efficiency of our method, we perform post-place and route dynamic timing analysis based on real operands extracted from a variety of applications. This helps to estimate the bit error rate, while considering the data dependent path excitation. When applied to an IEEE-754 compatible double precision Floating Point Unit (FPU), the proposed approach reduces the timing failures by 216.25× on average compared to the reference FPU design under an assumed 8.1% variation-induced worst-case path delay increase in a 45 nm process. Our results show that the path shaping alone introduces a negligible 0.25% area and 5.7% power overheads with no performance cost. Finally, we demonstrate that by combining the path shaping with aggressive operand bitwidth truncation, we enable power savings of up-to 44.7% due to the substantially reduced switching activity at minimal quality loss.**

***Index Terms*—Dynamic timing analysis (DTA), error-resilience, low-power FPU, operand truncation, path shaping, variation-aware design**

## I. Introduction

THE aggressive shrinking of transistor sizes has worsened process variations, which led to a 25% delay increase [1] and 20× higher leakage variation [2] in advanced nanometer technologies. These trends make circuits more prone to timing failures, thus threatening the circuit functionality, and hinder designs from meeting the targeted specifications [3]. Such delay variations further worsen under scaled voltages, which are considered necessary for saving power [4].

Manufacturers tend to adopt timing guardbands that force the circuit to operate at a lower frequency or a higher voltage, providing sufficient timing margins to mitigate any failures triggered by delay variations [2], [5]. However, such timing margins are considered to be overly pessimistic, since they are estimated based on few worst-case critical paths and on assumed rare operating conditions (e.g. temperature), and thus incurring large overheads [5], [3], [1], especially for the inherently fast paths.

In an attempt to trim down the introduced overheads, statistical static timing analysis (SSTA) tools have been introduced [6], but such tools still focus on improving the analysis and margin estimations rather than the design itself. Design-centric techniques focus on integrating extra circuits to detect any errors and either try to correct them in-situ using special flip-flops [7] or stall the pipeline and replay the failed instructions [8], [4]. Other design-centric schemes try to predict the instructions and operands that may activate the failure prone long latency paths (*LLPs*) [9] and provide extra clock cycle(s) for the completion of these paths. However, in all the cases the enforced timing constraints for timing-error detection and the overheads incurred by the applied recovery methods, especially when the activation probability of critical paths is high, may neglect the gains achieved by removing the static or dynamic safety margins. An approach proposed in [10] helps to limit the overheads of the above methods by reducing the overall number of failure prone critical paths. Although effective, such a method has never been applied to a fully pipelined design, nor has it been considered jointly with the above design-centric schemes or other methods to reduce the dynamic excitation of critical paths.

Recently, approximate computing has emerged as an alternative approach for addressing potential timing failures with less overheads than the ones incurred by the conventional guardband-based techniques [5], [11]. Existing studies have showcased the inherent resiliency of various signal/image processing [12], [13], machine learning [12], [14] and scientific computation algorithms [13] to faults or inaccurate operations. Most of the existing studies have indicated that any approximation should be applied only to error-resilient code or data regions in applications, since uniform approximation of all data may result in significant quality degradation [15], [16]. For instance, approximating few less significant bits (LSBs) in the mantissa part of floating point (FP) operands leads to insignificant quality loss as opposed to approximation of any bit of the exponent part of (FP) operands [17], [18]. The majority of approximation based schemes have been applied to standalone digital signal processing (DSP) accelerators or integer arithmetic units rather than to complex pipelined designs. Furthermore, such schemes have been evaluated on simulators using error-injection models that neglect the impact of any approximation, e.g. reduced precision, on the dynamic data dependent path excitation.

Few research studies tried to exploit the data dependent path excitation for limiting the static or dynamic margins by dynamically adjusting the clock period [19], [20]. Such schemes have shown the overly pessimistic estimation of timing paths by conventional static timing analysis (STA) compared to the dynamic timing analysis (DTA) approach that we propose here. However, the proposed adjustment of the clock per instruction may be very challenging to apply in practice, and has never been used in conjunction with approximation schemes.

**Contributions.** The primary objective of this paper is to minimize the potential timing failures in pipelined designs. In particular, we propose a method to redesign the target circuit for limiting the failure-prone *LLPs* and reduce the path excitation probability by truncating the operand bitwidth [21]. Our approach can effec-

tively reduce the overheads of traditional variation-aware schemes, while complementing existing design-centric and approximation-based schemes. The contributions of this work can be summarized as follows:

- We present a new framework to redesign a pipelined circuit by carefully shaping the path distribution such that the *LLPs* are significantly reduced and isolated to as few pipeline stages as possible. The advantages of such an approach are two-fold. First, it reduces the critical *LLPs* and thus significantly reduces the failure probability. Second, it limits the number of registers or stages, where any variation-aware or approximation scheme would need to be used, thus limiting any resulting overheads.
- We exploit the data dependent excitation of timing paths by truncating the bitwidth of the operands that may activate the few remaining *LLPs*. This is realized by setting a number of LSBs of these operands to a constant value of zero. By doing so, we reduce the computational delay of the *LLPs* along with the excitation probability of these paths. In addition, we also reduce the power consumption due to a lower switching activity. The operand bitwidth truncation may lead to a deterministic quality loss. However, we limit such loss by carefully selecting the number of truncated bits of the operands that may excite the isolated *LLPs*. Moreover, any loss incurred by truncation is expected to be less than the loss incurred by random timing failures, which may affect significant parts of the computation.
- We apply the proposed approach to the implementation of a variation-aware IEEE-754 compatible [22] Floating Point Unit (FPU) in a 45 nm process technology. Note that floating point variation-aware designs have not received much attention apart from few works [23], [24], despite the importance of FPUs in today's high-end processors. In addition, FPUs are excellent representatives of complex pipelined designs and any approximation scheme has a measurable effect on accuracy. By combining a set of constraints at the synthesis and place and route phases with micro-architectural changes, we show that the *LLPs* can be significantly reduced compared to an original FPU designed with conventional performance-centric optimizations.
- We present a post-layout DTA tool to estimate the dynamic excitation of timing paths and potential timing failures, considering the operands of executed program traces and the clock period.
- We develop a profiling tool to extract FP instruction traces and operands from various applications executed on a `RISC` (Reduced Instruction Set Computer) processor. We use the profiling results as input to the developed DTA tool for realistic evaluation of the efficacy of our approach.
- We estimate the timing failures under various assumed clock reduction (CR) levels that represent potential variations for the proposed and original designs under different operand bitwidth truncation ranges. Based on these estimates, we evaluate the quality loss quantified in terms of the Relative Error (RE) for four popular applications `Heartwall` `Raytrace`, `CFD` and `K-means`. These applications originate from the Medical Imaging, Computer Graphics, Fluid Dynamics and Data Mining domains, respectively. Finally, we estimate the bit error rate (BER) and show how it varies across the bit positions and across applications in the conventional and proposed FPU designs.

The rest of the paper is organized as follows. Section II presents the proposed approach. Section III describes the implemented flow using state-of-the-art tools. In Section IV, we apply the proposed framework to the design of an IEEE-754 compatible FPU of a `RISC` processor. Section V presents our experimental results. Section VI discusses related work. Finally, we draw our conclusions in Section VII.

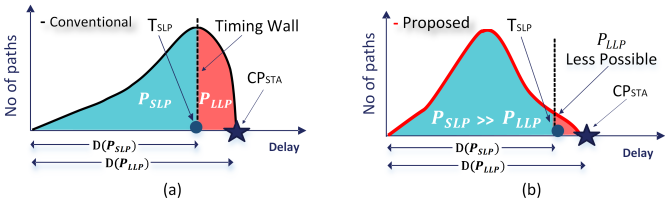


Fig. 1: (a) Conventional and (b) proposed path distribution.

## II. Proposed Approach

Let us consider a pipelined design which consists of a set of N unique combinational paths $P = \{p_1, p_2, ..., p_N\}$. In this design, each path completes with a delay $D(p_i)$ for $i = 1, 2, ..., N$. As in any synchronous design, the longest path across all $S$ pipeline stages determines the clock period, such as:

$$CP_{STA} = \max_{\mathbf{s=1....S}} \left\{ \max_{\mathbf{p \in P^s}} \{D(p)\} \right\} = \max_{\mathbf{p \in P}} \{D(p)\}, \quad (1)$$

where $P^s$ is the set of paths that belongs to pipeline stage $s$ ($s = 1, 2, ..., S$).

Figure 1a depicts a typical distribution of all the path delays $D(P)$, which is obtained by applying conventional design flows and STA. As it can be seen, such a distribution is characterized by a so-called "timing wall", with many *LLPs* close to $CP_{STA}$. Such a wall is a consequence of how modern designs are optimized for power and area, subject to a global frequency constraint. In particular, current design flows minimize the delay of *LLPs* by (area/power hungry) gate up-sizing, while the inherently short latency paths (*SLPs*) are allowed to become near-critical for recovering any area or power costs [10]. This "timing wall" does not have any negative impact on the adopted $CP_{STA}$ of the design, however it critically affects the probability of timing failures since under any (even small) delay variation many paths may fail [25].

Consider a set of $K$ *SLPs* as $P_{SLP} = \{p_1^{SLP}, ..., p_k^{SLP}\} \subset P$ and a set of operands $Op_{SLP}$ that excite such paths; and assume that all these paths can complete their computations within a time: $T_{SLP} = \max\{D(P_{SLP})\}$, as shown in Figure 1a. Figure 1a also depicts a set of $M$ *LLPs*, namely $P_{LLP} = \{p_1^{LLP}, ..., p_M^{LLP}\} \subset P$, which is activated by another set of operands $Op_{LLP}$. The execution of the *LLPs* can be completed within the $CP_{STA}$, but these paths need more time than $T_{SLP}$, i.e. $T_{SLP} < D(P_{LLP}) \leq CP_{STA}$. If at time instant $t$ the executed instruction that is in the pipeline stage $s$ activates only paths from $P_{SLP}$, then a positive timing slack ($CP_{STA} - D(p_i^{SLP})$) can be observed. Such a slack can be used as a safety margin against potential delay variations, minimizing the probability of a timing failure at that stage. Intuitively, by ensuring that only *SLPs* are being triggered across all stages, then the overall probability of a timing failure will be low.

### A. Path Shaping and Critical Stage Constraining

The first step of our approach for reducing the timing failures is to move away from a path distribution with many *LLPs* close to the $CP_{STA}$, which is typical in conventional designs (see Figure 1a). The goal of our work is to minimize $|P_{LLP}|$ subject to the the target clock period $CP_{STA}$ and power/area constraints.

To achieve such a goal, we define appropriate timing constraints for different path groups and impose them on the design during synthesis. By introducing such path constraints, we ensure that the inherently fast paths (i.e. *SLPs*) are not made slower, as opposed to the conventional approach. The end goal is to obtain

TABLE I: Quality degradation in terms of Relative Error (RE) under "random bit errors" induced by timing failures and "deterministic errors" induced by deliberately setting the last 32 bits of each operand to 0 in an FP addition.

| | Operand A (64 bits) | Operand B (64bits) | Output C (64bits) | Relative Error (RE) |
|---|---|---|---|---|
| Reference | 0100000010001111001111000000101<br>0001111010111000010100011110 1100<br>(decimal: 999.7525) | 0100000011010000000111101101 1110<br>0110011001100110011001100110 0110<br>(decimal: 16507.475) | 0100000011010001000110001100 1110<br>1000111101011100001010001111 0101<br>(decimal: 17507.2275) | 0 |
| "Random bit error" (Bit flip in the $10^{th}$ MSB) | 0100000010001111001111000000101<br>0001111010111000010100011110 1100<br>(decimal: 999.7525) | 0100000011010000000111101101 1110<br>0110011001100110011001100110 0110<br>(decimal: 16507.475) | 010000001**0** 0100010001100011001 1110<br>1000111101011100001010001111 0101<br>(decimal: ~1094.2017) | ~0.9375 |
| "Deterministic error" (truncating 32 LSBs) | 0100000010001111001111000000101<br>**00000000000000000000000000000000**<br>(decimal: 999.7524) | 0100000011010000000111101101 1110<br>**00000000000000000000000000000000**<br>(decimal: 16507.468) | 0100000011010001000110001100 1110<br>0010100000000000000000000000000<br>(decimal: ~17507.2211) | $\sim 3.6556 \cdot 10^{-7}$ |

a path distribution similar to the one depicted in Figure 1b, where $|P_{SLP}| >> |P_{LLP}|$. Note that to facilitate the isolation of discerned path groups, we also make modifications at the micro-architectural or register-transfer level (RTL) by trying to constrain the *LLPs* in as few stages as possible. This does not only help to control better the path groups, but also enables the isolation of $P_{LLP}$ to as many stages as accessed by only few specific instructions. This allows us to apply a failure mitigation or correction technique to few stages rather than using it for the whole design, which is far more complicated and costly [26]. In addition, it facilitates the development of failure mitigation mechanisms tailored for the specific instruction(s) that activate the few remaining *LLPs*. It is also important to note that our approach does not change the $CP_{STA}$, since any path shaping is made subject to maintaining the conventional speed.

### B. Significance-Driven Operand Truncation

The activation of each combinational path within each stage in a pipeline design depends on the executed instruction and on the input operands of each operation that takes place in each stage. Hence, it is possible to further reduce the activation probability of the *LLPs* by adjusting the operands of the specific instructions that excite these paths. A straightforward solution that we propose is to set to "0" some LSBs of $Op_{LLP}$ and truncate the bitwidth, thus reducing the computational delay. To elucidate the impact of bitwidth truncation on computational delay, let us consider a simple 4-bit ripple carry adder (RCA), as shown in Figure 2. The depicted adder consists of four full adders (FA). FA is a logic circuit that adds two input operand bits ($A$i, $B$i) plus a Carry in bit ($Ci,i$) and generates a Carry out bit ($Co,i$) and a sum bit ($S,i$). In such a design, the most timing critical path $LLP1$ (emphasized in red dotted line) will be activated when the carry propagates all the way from $C$i,0 to $Co$,3. If we define the gate delay as $T$, then $LLP1$ requires a delay equal to $8T$ to be completed, since $LLP1$ will have to travel from the AND gate (emphasized in red) in FA0 down to XOR gate (emphasized in red) in FA3. According to Eq. 1, the minimum $CP_{STA}$ of this datapath is $8T$. Note that such a

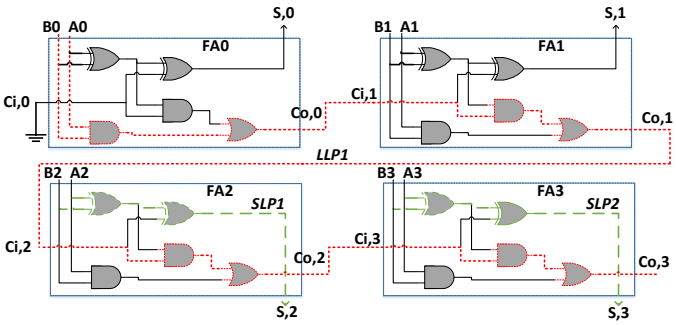



Fig. 2: Data-paths in a 4-bit ripple carry adder, highlighting the longest latency path (*LLP1*) and two short latency paths (*SLP1* and *SLP2*).

delay will be activated only in case of a suitable combination of operands belonging to $Op_{LLP}$. For instance, when $A = 1111$ and $B = 0001$, the carry generated in the first bit position (i.e. LSB) is propagated all the way to the final bit position, exciting the error-prone $LLP1$. Under an assumed variation induced delay increase this path will fail as $D(LLP1) > 8T$ and thus $D(LLP1) > CP_{STA}$. However, by modifying the inputs and inserting 0s in the last 2 bits, such as: $A = 1100$ and $B = 0000$, there is no carry propagation and thus only *SLPs* (e.g. $SLP1$ and $SLP2$ highlighted in green dashed lines) will be excited. By truncating the last 2 bits of the input operands the delay of the critical paths that are excited is reduced to $2T$, thus providing enough timing slack to address any potential delay increase.

The truncation of a number of LSBs from $Op_{LLP}$ may provide a slack and reduce the *LLPs* excitation as discussed above, but this will come at a quality loss. However, such a loss can be controlled by appropriately selecting the number of truncated LSBs, ensuring that it is not as catastrophic as the loss incurred by random timing failures when these affect the most significant bits (MSBs). For instance, let us consider the addition of two floating point operands A and B which results to an output C. These operands follow the IEEE-754 [22] double precision format in which the first bit from the left represents the sign, the next 11 bits represent the exponent, while the rest 52 bits represent the mantissa. As illustrated in Table I, a random error in the exponent part, e.g. in the $10^{th}$ bit of the output C (highlighted in red), induced by a timing failure will lead to a completely different number than the one expected resulting in high Relative Error of ~0.9375 (Relative Error is defined in Eq.2 in Section V.C,2). Conversely, in case of 32 LSBs truncation in the mantissa part of each operand, the resulting output value is very close to the reference value with very low RE equal to $\sim 3.6556 \cdot 10^{-7}$. Such a low RE is attributed to the fact that we truncate LSBs in the mantissa part that are not critical for determining the output value in FP operations. On the other hand, the exponent plays a significant role in determining the range of the output and any error either due to random bit flip or truncation in that part may result in catastrophic results [17].

### C. Exploiting the Dynamic Path Excitation

The fundamental difference between our approach and approaches that rely on STA, is that we exploit the dynamic excitation of paths by operands. By making the excitation of the *LLPs* by $Op_{LLP}$ rare by design, we minimize the need for adopting any conservative timing margin. In the case of $Op_{SLP}$ there is enough positive timing slack to avoid failures under any potential worst-case path delay increase up to a magnitude of:

$$\Delta T_{maxvar} \leqslant (CP_{STA} - T_{SLP})$$

To estimate the efficacy of our approach and evaluate the dynamic data dependent excitation of paths, we develop a tool to perform DTA. Such a tool allows us to explore the unused timing

margins of the processor that are available at runtime. These cannot be accurately characterized by STA due to the missing notion of path activation probabilities. Additionally, by applying this analysis phase, we estimate how often the *LLPs* are excited and the quality degradation incurred by operand truncation. Finally, such a DTA tool helps us extract instruction aware timing failures, which also depend on the dynamic excitation of critical paths by operands. The total number of timing failures and BERs can be extremely useful for more accurate error injection during the evaluation of application resiliency [27].

## III. Design Flow

The steps of the proposed approach are implemented using state-of-the-art electronic design automation (EDA) tools. The workflow of this approach is shown in Figure 3, where our modifications are highlighted in orange. The flow consists of a design phase and an analysis phase.

### A. Design Phase

To reduce $|P_{LLP}|$, we impose constraints on different path-groups (in the Synopsys Design Constraints or SDC file) based on the minimum delay required for the completion of each path-group. By introducing multiple path-group constraints, we force the synthesis tool to avoid optimizations that make naturally fast paths slower for saving area and power. Even though these constraints may affect area and power consumption, the total overheads, which depend on the targeted path distribution, can be kept small. Initially, we apply strict constraints to shift the "timing wall" away from the target $CP_{STA}$. If there are timing violations after synthesis, we relax the design constraints and re-run our iterative method until the timing target is met. The applied design constraints, which are implemented in a fully automated way, not only reduce the number of the timing critical paths, but also isolate them to as few pipeline stages as possible. However, the design can be further improved by introducing custom changes in the micro-architecture/RTL description, which helps the automated process of the path redistribution technique. After performing these changes, we have managed to restrict $P_{LLP}$ to one stage, whereas ensuring that the target clock frequency of the design is still met. The amount and type of these modifications depend on the desired path distribution and the original design. Nevertheless, these modifications may not be needed at all if the design facilitates a desired path distribution after imposing the group path constraints. In this work, we apply strict timing constraints, under iso-frequency, and the preferred path shaping can be achieved only after performing micro-architectural/RTL changes. Note that the same path distribution can be obtained without applying such micro-architectural changes, but this incurs considerable timing penalties. The synthesis step is followed by the place and route using the Innovus tool from Cadence. Sign-off STA follows with Synopsys PrimeTime to verify that design has achieved the timing closures.

### B. Analysis Phase

*1) DTA:* To enable characterization of the data dependent path activation, we use the post-place gate-level simulation supported by ModelSim, which monitors the inputs and the outputs of all flip-flops in the design and generates a corresponding event log. To obtain this information and perform full back annotated simulation, ModelSim apart from RTL netlist and testbench requires a standard delay format (SDF) file which describes the cell and interconnect delay. RTL netlist and SDF file are obtained at the place and route step.

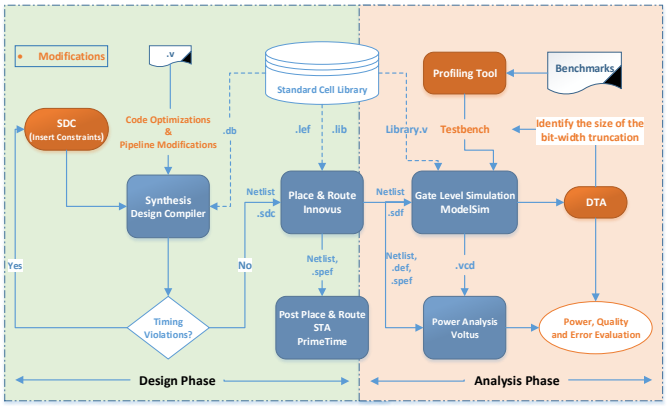



Fig. 3: Workflow of the proposed approach. Our modifications are emphasized in orange.

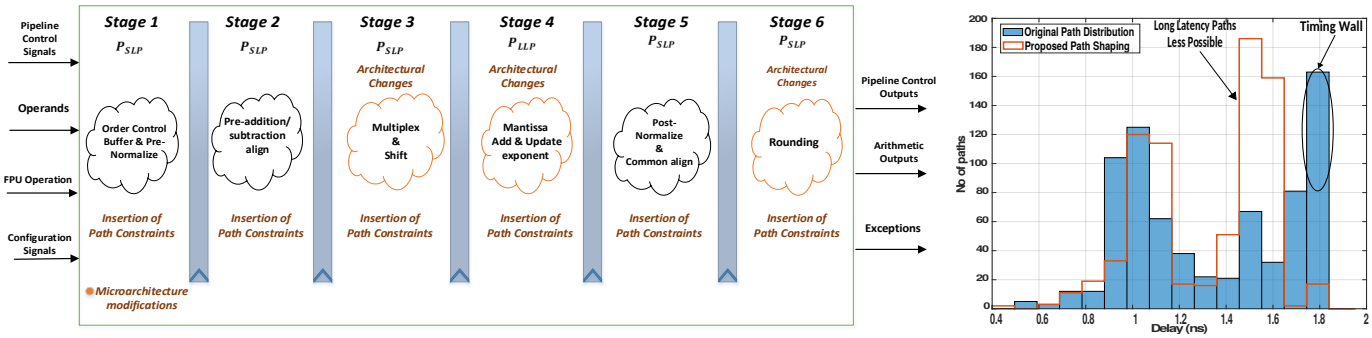
*2) Profiling Tool:* To feed the ModelSim with real FP operands, we extended a sampling-based profiling tool [28] to collect statistics about operands of FP instructions running different applications on a real hardware. The tool consists of an online module, that runs in parallel with a profiled application, and an offline module used to process the profiling results after the application has finished. The online module interrupts the execution of a running program with a defined period, retrieves the current instruction pointer and collects the values held in registers. The offline module disassembles the profiled program to identify types of instructions executed by the application. It uses this information to assign the sampled registers to specific instructions using the collected instruction pointers. At the final stage of profiling, we filter all sampled instructions to find all FP-instructions and values held in registers used by these instructions. To implement our tool, we use *ptrace* interfaces which are supported by the latest Linux kernel (since Linux 2.6.34).

Provided that every set of the extracted operands under nominal conditions produces an error-free output, we define this error-free gate-level simulation output as $D_{gold}$. To measure the number of manifested timing failures under any potential delay increase, we execute the simulation and compare $D_{gold}$ to the event log obtained from ModelSim. Finally, this tool extracts a value change dump (VCD) file that contains information about the switching activity and value changes that occurred during the simulation for nets and registers of the design. This file is essential for performing dynamic power analysis.

*3) Power Analysis:* We estimate the power consumed by all benchmarks and designs using the Voltus tool from Cadence. To perform dynamic power analysis, we use the following input data in Voltus: the post placed and routed netlist, the VCD file, a design exchange format (DEF) file that represents the physical layout and a standard parasitic exchange format (SPEF) file which corresponds to the parasitic data of wires in a chip. Sign-off quality power analysis obtains VCD files from ModelSim, while the other inputs are produced by the already placed and routed design.

## IV. Case Study: Application to FPU

We apply the proposed approach to a multi-cycle, IEEE-754 compatible double precision FPU. According to the IEEE-754 Standard, a FP number follows the representation $-1^S \times M \times 2^E$, where S : sign, E : exponent and M : mantissa. In a double precision FP number the MSB indicates the sign, the next 11 bits represent the exponent and the mantissa consists of the 52 LSBs. This FPU is a part of the latest Out-of-Order mor1kx MAROCCHINO pipeline, a



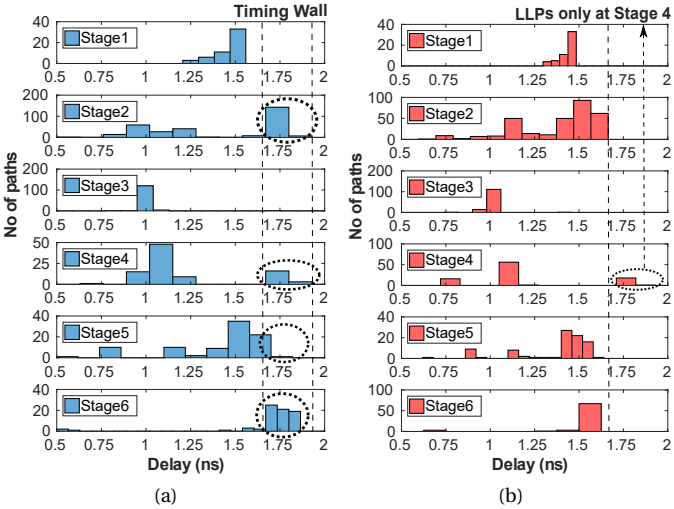

(a) (b)

Fig. 4: (a) Proposed micro-architecture of the FP addition/subtraction related stages, highlighting our modifications in orange and (b) path distribution of the original and proposed FPUs.

5-stage pipeline microprocessor based on the `OpenRISC` 1000 Instruction Set Architecture [29]. In this paper, we implement the following FP instructions: addition/subtraction, integer-to-FP and FP-to-integer conversions and comparison of FP operands. Figure 4a illustrates the micro-architecture of the targeted FPU, highlighting the FP addition/subtraction. At Stage 1, an Order Control Buffer and a Pre-Normalize block are implemented, which permits data dependencies detection and adjustment of the exponent and mantissa, respectively. Stage 2 is responsible for the pre-addition/subtraction alignment, while Stage 3 performs the necessary multiplexing and shifting of the operands. Mantissa addition and exponent update are performed at Stage 4; rounding occurs in the last two stages.

### A. Redesigned FPU

We start by applying the typical EDA flow (see Figure 3) to the conventional unmodified design. After following the synthesis and place and route steps, as well as performing STA using PrimeTime, we built the path distribution that is shown in Figure 4b. The obtained distribution implies that the conventional performance-centric flow results in a large $|P_{LLP}|$, in which many paths are close to the worst case delay, or, in other words, to the clock delay. Such a path distribution creates the "timing wall". Figure 5a depicts the path distribution within each pipeline stage, revealing that the "timing wall" exists in 4 out of the 6 stages. These findings




(a) (b)

Fig. 5: (a) Original and (b) proposed path distribution across the six pipeline stages.

indicate that there is an increased probability of timing failures in the stages where *LLPs* exist. To circumvent this, we apply the steps of our design flow with the following modifications:

*1) Micro-architectural changes:* The way RTL is written has a direct impact on the physical layout and thus on the path distribution. To change the distribution of paths in the way discussed in Section II, we apply an extra optimization step that focuses on modifying the micro-architecture/RTL description of some pipeline stages. In particular, after performing STA, we noticed that Stage 4 consists of the most timing critical paths. To reduce the number of the *LLPs* in this stage, we moved parts of the combinational logic to the previous stage, exploiting the slack margins observed at Stage 3. Additionally, rounding, which occurs at Stages 5 and 6, poses a bottleneck because it is applied to the result of all the FP operations, and thus many paths in these stages have a long latency. To this end, we changed the RTL code implementing the logic in Stages 5 and 6, making the synthesis part more efficient. Specifically, we optimized some unnecessary large conversions to negative numbers.

*2) Group path constraints:* As discussed in Section II.A,1, during the synthesis step, we apply various constraints by grouping paths into two different sets, the $P_{SLP}$ and the $P_{LLP}$, based on their computational delays. We define the delay target of the *SLPs* as $T_{SLP}$. If the path delay is less than $T_{SLP}$, then this path is assigned to the $P_{SLP}$, otherwise it is assigned to the $P_{LLP}$. Initially, we synthesize the design for a small $T_{SLP}$ in order to move paths away from $CP_{STA}$. If after synthesis the timing target is not met, then we increase the value of $T_{SLP}$ until the design achieves the targeted timing closures. After many iterations we set the $T_{SLP}$ in the particular FPU to 1.68ns. These iterations are implemented using tool command language (tcl) scripts. As a result, this automated procedure reduces $|P_{LLP}|$ (see Figure 4b), while ensuring all other paths are fast enough (at least 9.1% faster than the $CP_{STA}$) to tolerate variation-induced timing failures. Note that the timing constraints imposed by our design does not exceed the $CP_{STA}$, which is determined by the original unmodified design. We also constrain the *LLPs* to as few stages and instructions as possible. It is also worth mentioning that the remaining *LLPs* are isolated in such a way that can be triggered only by FP addition/subtraction instructions at Stage 4, which is the main goal of our design strategy.

### B. Application of Operand Truncation

After redesigning the FPU, we apply bitwidth truncation to the input operands of the specific instructions that activate the few remaining *LLPs* in order to reduce their excitation probability and

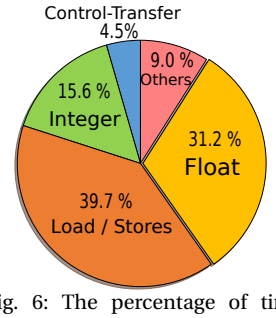



Fig. 6: The percentage of time spent on execution of different types of instruction.

| CR | Clock (ns) | Reduction (%) |
|---|---|---|
| CR0 | 1.85 | 0 |
| CR1 | 1.8 | 2.7 |
| CR2 | 1.75 | 5.4 |
| CR3 | 1.7 | 8.1 |
| CR4 | 1.65 | 10.8 |

TABLE II: Clock Reduction (CR) levels that represent potential worst-case delay increase.

thus timing failures. Since all the *LLPs* are restricted to Stage 4 of the FP addition/subtraction instructions, we deploy the truncation only to the LSBs of the input operands of these instructions. Given that Stage 4 implements the 52-bit mantissa addition, we set constant "0" values to the LSBs to avoid delay failures in the MSBs.

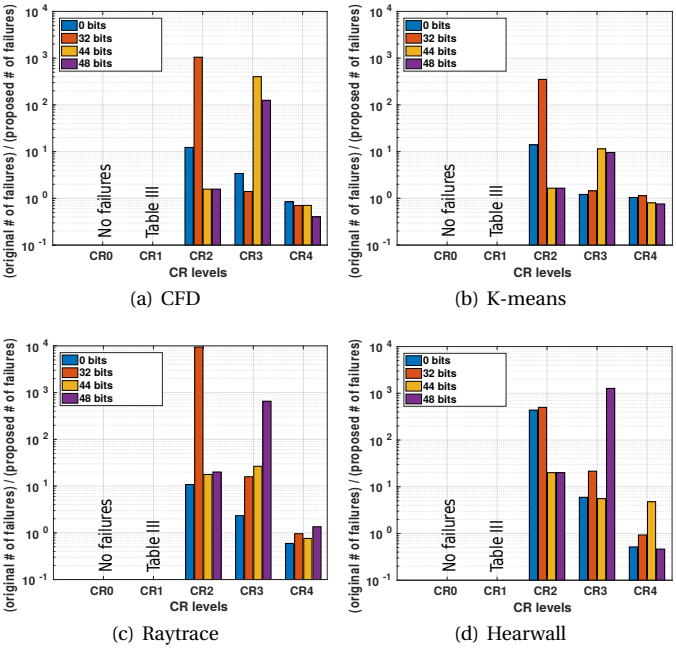
As discussed in Section II, truncation may lead to quality loss and needs to be carefully selected. In our experiments, we evaluated the truncation of the 32, 44 and 48 LSBs of the mantissa part of the FP addition/subtraction operands. This means that the sign and the exponent part of the IEEE-compliant operands were unaffected (12 bits totally) along with the 20, 8 and 4 MSBs of the mantissa part in the considered scenarios.

## V. Evaluation Results

In this Section, we evaluate the efficacy of our approach in limiting the timing failures under various CR levels, which are shown in Table II. We reduce the clock period from 1.85ns to 1.65ns in steps of 0.05ns, representing potential degrees of worst-case path delay increase that may be caused by variations [30]. In this Section, we compare our redesigned FPU (see Section III.A) with the original (reference) design. For a fair comparison, we applied the truncation of 32, 44 and 48 LSBs of specific FP operands to the original (Orig), unmodified FPU, and also to the proposed (Prop) one. The considered FPUs are implemented using the design flow described in Section II.A in NanGate 45 nm Composite Current Source Cell Library [31].

### A. Application Profiling

To estimate the efficacy of our approach using real FP operands (Section III.B,2), we developed a profiling tool to extract program traces from various compute intensive applications. In our analysis, we use the `Kmeans`, `CFD`, `Heartwall` from the Rodinia benchmark suite and the `Raytrace` benchmark from the Parsec suite. This set of benchmarks represents a variety of algorithms that have many FP operations and covers a wide range of domains, i.e. Data Mining, Fluid Dynamics, Medical Imaging and Computer Graphics. To obtain the program traces, we profile all benchmarks on an `ARM A7` based system, Odroid-Xu3. Figure 6 depicts the percentage of time spent on execution of different types of instructions averaged over all profiled benchmarks. We observe that benchmarks spent 31.2% of the total time on execution of FP instructions on average, which indicates their importance. Using such a tool, we extract 10000 operands from the most frequently executed FP instructions for each application, which we feed to the DTA tool to estimate the BER and the consumed power.




(a) CFD (b) K-means

(c) Raytrace (d) Hearwall

Fig. 7: Number of timing failures in the original FPU divided by the number of timing failures in the proposed FPU under various Clock Reduction (CR) and operand truncation levels across 4 benchmarks (legend shows the number of the truncated bits).

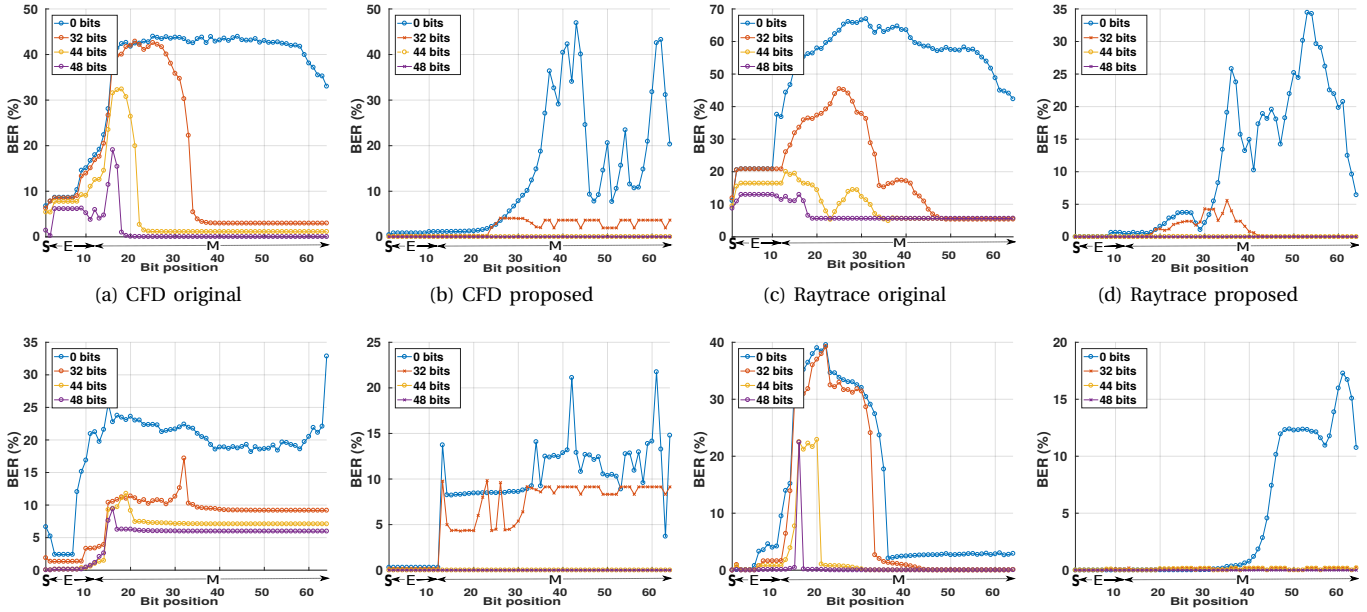
### B. Characterization of Timing Failures

Timing failures are a function of input operands, clock period and the number of truncated bits. Figure 7 demonstrates how the number of timing failures changes under different bitwidth truncation levels when we scale down the clock period in the original and the proposed designs across the 4 benchmarks. Note that the failures are reported only for the cases when the simulated output for specific operands differs from the recorded error-free output $D_{gold}$. As shown in Figure 7, under the nominal clock period (CR0) no failures are manifested. Moreover, we observe that the timing failures are substantially lower for any degree of bitwidth truncation across all applications. In particular, by setting to zero the 32, 44 and 48 LSBs of the mantissa in the original FPU, the total number of failures across the 4 benchmarks is reduced by 1.69×, 2.83× and 12.13× on average.

In the same figure, we observe that the number of timing failures incurred in the original design is significantly higher than the number of failures incurred in the proposed FPU under CR1, CR2 and CR3. Beyond the CR3 level, which corresponds to the $\Delta T_{maxvar}$ (Section II.C), we notice an increase in timing failures, especially in the proposed design. This can be attributed to the fact that the applied path shaping has shifted the paths, as shown

TABLE III: Timing failures under CR1 and various truncation levels across all benchmarks in case of the original (Orig) and the proposed (Prop) designs

| Number of truncated bits | benchmarks | | | |
|---|---|---|---|---|
| | CFD | Raytrace | K-means | Heartwall |
| 0 (Orig , Prop) | 107505 , 0 | 79985 , 0 | 16058 , 0 | 1101 , 0 |
| 32 (Orig, Prop) | 24508 , 0 | 38 , 0 | 78 , 0 | 32 , 0 |
| 44 (Orig, Prop) | 86 , 0 | 38 , 0 | 78 , 0 | 22 , 0 |
| 48 (Orig , Prop) | 86 , 0 | 22 , 0 | 78 , 0 | 22 , 0 |




(a) CFD original (b) CFD proposed (c) Raytrace original (d) Raytrace proposed

(e) K-means original (f) K-means proposed (g) Hearwall original (h) Heartwall proposed

Fig. 8: Bit error rate (BER) under CR3 and various operand bitwidth truncation levels across all benchmarks for the original and proposed designs. Legend shows the number of the truncated bits, while X axis depicts the sign (S) bit, the exponent (E) bits and the mantissa (M) bits.

in Figure 4b. This results in a high number of paths being activated and failing under CR4. In particular, Figure 4b shows that there are many paths with delay more than 1.65 ns in the original and proposed designs; this outcome implies that the probability of timing violations in case of activation of these paths under CR4 is expected to be very high. Note that this is a choice for the specific implementation and can be altered at design time.

Nonetheless, the proposed design results in a significantly reduced number of timing failures for most of the CR levels. Specifically, our approach under CR1 allows us to eliminate all the incurred errors, while Table III shows the original FPU exhibits from 86 up-to 107505 failures for the same CR. In addition, under CR3, which corresponds to an 8.1% worst-case path delay increase, the proposed design with the preferred path shaping reduces the number of timing failures by $\sim 4\times$ on average compared to the original design. The combination of path shaping and operand bitwidth truncation in the 32, 44 and 48 LSBs reduces these failures by $\sim 13\times$, $\sim 79\times$ and $\sim 769\times$ on average, respectively, when compared to the original design.

### C. Evaluation of BER and Quality

*1) BER:* Figure 8 depicts the average BER across the 4 benchmarks at CR3, where several interesting observations can be made. Note that in this figure letters S, E and M on the X axis correspond to the sign bit, the exponent bits and the mantissa bits, respectively. To begin with, we observe that distinct bit positions incur different BERs. This happens because different input operands activate different paths contributing to the calculation of different output bits. We also observe that the original design with the full range precision (0 bits truncation) exhibits high BERs, ranging from 1.15% up-to 66.9% in the mantissa and up-to 37.6% in the exponent part. To reduce BERs, we set different LSBs of the mantissa to "0", which is expected to reduce the number of the *LLPs* excitation and consequently the overall BERs. However, upon further investigation we observe that in the original FPU, the BER of the MSBs of the mantissa still remains considerably high under different truncation levels ($\sim 30\%$, $\sim 20\%$ and $\sim 15\%$ on average after truncating 32bits, 44bits and 48bits, respectively), while the BER of the exponent bits ranges from 0% up-to 20%. In contrast to the original FPU, the proposed FPU accompanied with path shaping and operand truncation results in significantly lower BERs across the vast majority of the bits and especially in the exponent (0% - 0.3%) and the MSBs of the mantissa (0% - 9%). The fact that the failing bits are neither in the exponent bits nor in the MSBs of the mantissa helps limit the incurred quality loss.

*2) Quality:* : To evaluate the quality loss incurred by the random timing failures and the operand truncation, we estimate the average Relative Error (RE) achieved by our design and compare it with the RE of the original design for all considered applications. The average Relative Error, a common metric for estimating the output quality [17], [32], is defined as:

$$RE = \frac{\sum_{i=1}^{K} \left| \frac{D_{gold}(i) - O_{sim}(i)}{D_{gold}(i)} \right|}{K} \quad (2)$$

,where $D_{gold}$(i) denotes the exact error-free output value obtained from the reference FPU design and $O_{sim}$(i) represents the output obtained by the simulation using our DTA tool for a specific ($i$) FP instruction and the associated operands. The $O_{sim}$(i) value is extracted by the output register of the considered FPU after simulating both designs (original and proposed) under a specific CR level and bitwidth truncation range. For these experiments we extracted 10000 FP instructions for each benchmark and thus $i$ varies from 1 up-to $K$ = 10000. The effect of different CR and operand bitwidth truncation levels on the RE before and after applying our approach is depicted in Figure 9. The original design without any truncation under the nominal clock period (CR0) introduces no quality degradation since no failures have been manifested (see Figure 7). In the case of `CFD`, we can observe that the RE of the original FPU leads to unacceptable (> 1) RE levels even under a small worst-case delay increase (i.e. CR1). Truncation helps to reduce the RE under CR1, CR2 and CR3, but not for every CR and truncation level. On the other hand, the proposed FPU, under the same CR levels renders the quality loss controllable and deterministic, since it depends only on the number of LSBs that

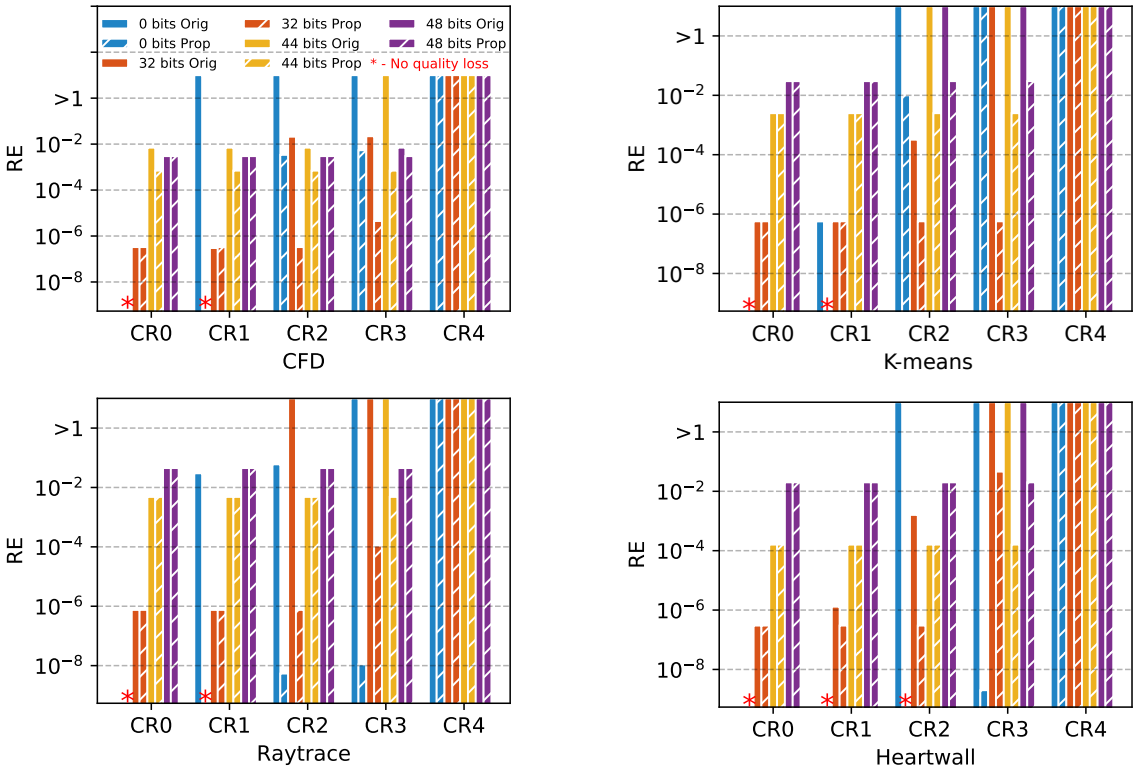



Fig. 9: Average Relative Error (RE) under various Clock Reduction (CR) and operand bitwidth truncation levels across all benchmarks for the original (Orig) and the proposed (Prop) designs (legend shows the number of the truncated bits).

are truncated. In the case of `Raytrace`, we obtain similar results as the proposed design under CR3 achieves up-to 0.003 RE, while the RE incurred by the original design may reach unacceptable values. In the case of `Kmeans` and `Heartwall`, we observe that the original FPU under CR3 and all the considered truncation levels incurs a significant quality degradation. Conversely, the proposed FPU under CR3 exhibits RE ranging from $\sim 2{\cdot}10^{-9}$ up-to $\sim 0.04$.

Overall, the proposed FPU minimizes the catastrophic quality loss incurred by the original double precision FPU. When compared to the original FPU with various truncation levels enabled, the combination of path shaping and operand truncation provides equal or much lower REs for the first four CR levels across different benchmarks. Beyond the CR3 level, we notice a significant quality degradation in the original and the proposed designs. This can be explained by a high likelihood of massive timing failures under CR4, as discussed in Section V.B. Finally, we observe that setting the last 32 bits of mantissa to "0" provides a judicious choice for operand truncation, considering all the evaluated applications.

### D. Estimation of Power and Area

Table IV depicts power consumed by the post-placed and routed original and proposed designs measured with the EDA tools, as explained in Section II.A.2. Our approach introduces a 5.7% power overhead when path shaping is applied. However, the proposed FPU leads to up-to 44.7% power savings when path shaping and 48 LSBs truncation are combined due to a significantly reduced switching activity. Overall, the area overhead incurred by path shaping is 0.25%, while our approach does not incur any performance loss, which would otherwise have been incurred by conventional guardband based techniques.

### E. Discussion on Potential Power Savings

We propose a variation-aware approach that facilitates the positive timing slacks of the *SLPs* and exploits the rare activation of the *LLPs* through significance-driven bitwidth truncation. Conversely, such properties could also be utilized to allow operation at a reduced voltage, when the manufactured chip is unaffected by variations. In this case, our design can facilitate operation at 0.95V, leading to extra $\sim$28% power gain on average compared to the operation at the nominal supply voltage of 1.1V. This means that the proposed FPU operating at 0.95V enables us to save 72.7% of power on average in the case of 48 *LSBs* truncation compared to the original FPU operating at 1.1V. In other words, our approach can be used not only for mitigating timing failures at a low cost, but also for enabling operation at a reduced voltage.

In this paragraph, we compare our design with the guardband based approach. In particular, according to the conventional paradigm, the original FPU adopts enough timing guardbands by scaling up the voltage to avoid failures and obtain an error-free output. Using the available fast corner cell library (@1.25V) and the extracted VCD files, which allow us to consider the

TABLE IV: Average power savings in the proposed design under various truncation levels. Note that the average power dissipation of the original (unmodified) design under iso-voltage is 24.71 mW.

| Number of truncated bits | Power consumption in the proposed FPU (mW) | Power Savings (%) |
|---|---|---|
| 0 | 26.12 | -5.7 |
| 32 | 19.99 | 19.1 |
| 44 | 16.91 | 31.6 |
| 48 | 13.66 | 44.7 |

switching activity and dynamic path activation, we estimate the power consumption across all applications. Operation at such a voltage provides the necessary margins to avoid all the dynamic timing failures. However, as shown in Table V, it comes at a cost of up-to 43.1% power overhead when compared to the proposed design (@1.1V) without enabling the operand truncation. If we combine the path shaping technique with the 48 *LSBs* operand truncation, which also leads to an error free output (see Figure 8), our approach can lead to 84.21% power savings on average.

TABLE V: Power savings across all the benchmarks in our design when only path shaping is enabled compared to a guardband based FPU

| Benchmarks | K-means | CFD | Heartwall | Raytrace |
|---|---|---|---|---|
| Power gains (%) | 40.52 | 35.76 | 43.11 | 38.64 |

## VI. Related Work

Our work aims at preventing timing failures by truncating the bitwidth of operands, while avoiding the use of conservative guardband based schemes. The majority of the works in the area of approximate computing exploited the inherent error resilience of applications to tolerate failures [33] and/or limit the overheads incurred by guardbands in at least some parts of the application [34]. However, such approaches have been evaluated on simulators due to the lack of suitable approximate hardware. Some of the existing approaches exploited the precision scaling to reduce the power consumption as in [35]. [36] accommodates voltage scaling in arithmetic units for saving energy by disabling some of the input bits. Another scheme proposed in [37] prune the bitwidth of all the input operands in simple data-paths, enabling power savings due to the reduced switching activity. Although very interesting, such works overlook the impact of precision scaling on the reduction of the path-delay; and did not consider 'when and where' they need to apply it in pipelined datapaths as we do.

Some works attempted to exploit an unequal contribution of algorithmic computations to output quality. In particular, several works proposed to redesign the circuit by giving priority to the execution of the most critical parts of each datapath [16] for avoiding failures. These works may have exploited the varying importance of computations, but they did so only for few application specific DSP architectures rather than for general purpose FPUs that may run any application.

Some recent works used approximate computing to make circuits resilient to variation-induced delay failures. An approach in [38] uses precision scaling to limit timing failures in the context of transistor aging and thus mitigating the estimated delay increase over few years. A post-Silicon technique in [18] truncates the bitwidth of all the inputs operands to prevent timing failures in typical DSP hardware modules.

Overall, the majority of the proposed solutions are applied to simple data-paths or arithmetic units rather than complex pipelined designs. Moreover, existing schemes reduce the computation precision neglecting the fact that there are few operands and instructions that may activate the error-prone *LLPs*. The novelty of the proposed significance-driven operand truncation technique lies in applying bitwidth truncation only to the error-prone operands of specific instructions that have been isolated by design.

## VII. Conclusion

This paper presents a framework for minimizing the timing failures in pipelined designs by i) redesigning the target circuit in a way that eliminates the excitation of the *LLPs*, and ii) opportunistically exploiting the dynamic activation of such paths by few operands $Op_{LLP}$ and making them rare by setting a constant value "0" to a fixed number of LSBs in the relevant operands. The evaluation of the proposed redesigned placed and routed FPU with the developed DTA tool using extracted program traces shows a significant reduction of timing failures under potential delay variations with a negligible 0.25% area overhead and no performance loss. An essential attribute that led to low overheads in our design is the applied path shaping technique. Without this scheme the truncation should be applied statically to every operation and operand. Our results also show that the proposed approach effectively reduces the BER in all the exponent and mantissa bits of the redesigned FPU, as opposed to the reference design. Truncation of 32 or 44 LSBs of $Op_{LLP}$ helps to maintain low RE levels in all the evaluated applications and up-to an assumed 8.1% variation-induced worst-case delay increase. Finally, we observe that path shaping may introduce 5.7% power overhead, but when combined with operand bitwidth truncation can lead to up-to 44.7% power savings. Even though we demonstrated the efficacy of the proposed approach by applying it to the specific FPU, the presented steps can be applied to redesign the stages of any other pipelined core.

## Acknowledgments

The presented research effort is partially supported by the European Community Horizon 2020 programme under grant no. 688540 (UniServer) and grant no. 732631 (OPRECOMP).

## References

[1] P. Gupta, Y. Agarwal, L. Dolecek, N. D. Dutt, R. K. Gupta, R. Kumar, S. Mitra, A. Nicolau, T. S. Rosing, M. B. Srivastava, S. Swanson, and D. Sylvester, "Underdesigned and opportunistic computing in presence of hardware variability," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 8–23, 2013. doi: 10.1109/TCAD.2012.2223467

[2] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th Design Automation Conference, DAC 2003, Anaheim, CA, USA, June 2-6, 2003*. ACM, 2003. doi: 10.1145/775832.775920 pp. 338–342.

[3] K. A. Bowman, J. W. Tschanz, S. Lu, P. A. Aseron, M. M. Khellah, A. Raychowdhury, B. M. Geuskens, C. Tokunaga, C. Wilkerson, T. Karnik, and V. K. De, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *J. Solid-State Circuits*, vol. 46, no. 1, pp. 194–208, 2011. doi: 10.1109/JSSC.2010.2089657

[4] D. M. Bull, S. Das, K. Shivashankar, G. S. Dasika, K. Flautner, and D. T. Blaauw, "A power-efficient 32b ARM ISA processor using timing-error detection and correction for transient-error tolerance and adaptation to PVT variation," in *IEEE International Solid-State Circuits Conference, ISSCC 2010, Digest of Technical Papers, San Francisco, CA, USA, 7-11 February, 2010*. IEEE, 2010. doi: 10.1109/ISSCC.2010.5433919 pp. 284–285.

[5] G. Karakonstantis, A. Chatterjee, and K. Roy, "Containing the nanometer "pandora-box": Cross-layer design techniques for variation aware low power systems," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 1, no. 1, pp. 19–29, 2011. doi: 10.1109/JETCAS.2011.2135590

[6] C. S. Amin, N. Menezes, K. Killpack, F. Dartu, U. Choudhury, N. Hakim, and Y. I. Ismail, "Statistical static timing analysis: how simple can we get?" in *Proceedings of the 42nd Design Automation Conference, DAC 2005, San Diego, CA, USA, June 13-17, 2005*, W. H. J. Jr., G. Martin, and A. B. Kahng, Eds. ACM, 2005. doi: 10.1145/1065579.1065751 pp. 652–657.

[7] Y. Zhang, M. Khayatzadeh, K. Yang, M. Saligane, N. R. Pinckney, M. Alioto, D. T. Blaauw, and D. Sylvester, "irazor: Current-based error detection and correction scheme for PVT variation in 40-nm ARM cortex-r4 processor," *J. Solid-State Circuits*, vol. 53, no. 2, pp. 619–631, 2018. doi: 10.1109/JSSC.2017.2749423

[8] D. T. Blaauw, S. Kalaiselvan, K. Lai, W. Ma, S. Pant, C. Tokunaga, S. Das, and D. M. Bull, "Razor II: in situ error detection and correction for PVT and SER tolerance," in *2008 IEEE International Solid-State Circuits Conference, ISSCC 2008, Digest of Technical Papers, San Francisco, CA, USA, February 3-7, 2008*. IEEE, 2008. doi: 10.1109/ISSCC.2008.4523226 pp. 400–401.

[9] S. Ghosh, D. Mohapatra, G. Karakonstantis, and K. Roy, "Voltage scalable high-speed robust hybrid arithmetic units using adaptive clocking," *IEEE Trans. VLSI Syst.*, vol. 18, no. 9, pp. 1301–1309, 2010. doi: 10.1109/TVLSI.2009.2022531

[10] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Slack redistribution for graceful degradation under voltage overscaling," in *Proceedings of the 15th Asia South Pacific Design Automation Conference, ASP-DAC 2010, Taipei, Taiwan, January 18-21, 2010*. IEEE, 2010. doi: 10.1109/ASPDAC.2010.5419690 pp. 825–831.

[11] S. Mittal, "A survey of techniques for approximate computing," *ACM Comput. Surv.*, vol. 48, no. 4, pp. 62:1–62:33, 2016. doi: 10.1145/2893356

[12] H. Zhang, M. Putic, and J. Lach, "Low power GPGPU computation with imprecise hardware," in *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*. ACM, 2014. doi: 10.1145/2593069.2593156 pp. 99:1–99:6.

[13] B. Grigorian, N. Farahpour, and G. Reinman, "BRAINIAC: bringing reliable accuracy into neurally-implemented approximate computing," in *21st IEEE International Symposium on High Performance Computer Architecture, HPCA 2015, Burlingame, CA, USA, February 7-11, 2015*. IEEE Computer Society, 2015. doi: 10.1109/HPCA.2015.7056067 pp. 615–626.

[14] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," in *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2012, London, UK, March 3-7, 2012*, T. Harris and M. L. Scott, Eds. ACM, 2012. doi: 10.1145/2150976.2151008 pp. 301–312.

[15] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan, "Axnn: energy-efficient neuromorphic systems using approximate computing," in *International Symposium on Low Power Electronics and Design, ISLPED'14, La Jolla, CA, USA - August 11 - 13, 2014*, Y. Xie, T. Karnik, M. M. Khellah, and R. Mehra, Eds. ACM, 2014. doi: 10.1145/2627369.2627613 pp. 27–32.

[16] G. Karakonstantis, N. Banerjee, and K. Roy, "Process-variation resilient and voltage-scalable DCT architecture for robust low-power computing," *IEEE Trans. VLSI Syst.*, vol. 18, no. 10, pp. 1461–1470, 2010. doi: 10.1109/TVLSI.2009.2025279

[17] W. Liu, L. Chen, C. Wang, M. O'Neill, and F. Lombardi, "Design and analysis of inexact floating-point adders," *IEEE Trans. Computers*, vol. 65, no. 1, pp. 308–314, 2016. doi: 10.1109/TC.2015.2417549

[18] S. Narasimhan, K. Kunaparaju, and S. Bhunia, "Healing of DSP circuits under power bound using post-silicon operand bitwidth truncation," *IEEE Trans. on Circuits and Systems*, vol. 59-I, no. 9, pp. 1932–1941, 2012. doi: 10.1109/TCSI.2011.2180447

[19] J. Constantin, L. Wang, G. Karakonstantis, A. Chattopadhyay, and A. Burg, "Exploiting dynamic timing margins in microprocessors for frequency-over-scaling with instruction-based clock adjustment," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*, W. Nebel and D. Atienza, Eds. ACM, 2015. doi: 10.7873/DATE.2015.0303 pp. 381–386.

[20] A. Rahimi, L. Benini, and R. K. Gupta, "Application-adaptive guardbanding to mitigate static and dynamic variability," *IEEE Trans. Computers*, vol. 63, no. 9, pp. 2160–2173, 2014. doi: 10.1109/TC.2013.72

[21] I. Tsiokanos, L. Mukhanov, D. S. Nikolopoulos, and G. Karakonstantis, "Minimization of timing failures in pipelined designs via path shaping and operand truncation," in *24th IEEE International Symposium on On-Line Testing And Robust System Design, IOLTS 2018, Platja D'Aro, Spain, July 2-4, 2018*, D. Gizopoulos, D. Alexandrescu, M. Maniatakos, and P. Papavramidou, Eds. IEEE, 2018. doi: 10.1109/IOLTS.2018.8474084 pp. 171–176.

[22] IEEE 754-2008. IEEE 754-2008 Standard for Floating-Point Arithmetic.

[23] X. Liang, D. M. Brooks, and G. Wei, "A process-variation-tolerant floating-point unit with voltage interpolation and variable latency," in *2008 IEEE International Solid-State Circuits Conference, ISSCC 2008, Digest of Technical Papers, San Francisco, CA, USA, February 3-7, 2008*. IEEE, 2008. doi: 10.1109/ISSCC.2008.4523228 pp. 404–405.

[24] S. Salehi and R. F. DeMara, "Energy and area analysis of a floating-point unit in 15nm cmos process technology," in *SoutheastCon, April*, 2015. doi: 10.1109/SECON.2015.7132972 pp. 1–5.

[25] J. Patel, "Cmos process variations: A critical operation point hypothesis," April 2008 [Online].

[26] I. Tsiokanos, L. Mukhanov, D. S. Nikolopoulos, and G. Karakonstantis, "Variation-aware pipelined cores through path shaping and dynamic cycle adjustment: Case study on a floating-point unit," in *Proceedings of the International Symposium on Low Power Electronics and Design, ISLPED 2018, Seattle, WA, USA, July 23-25, 2018*. ACM, 2018. doi: 10.1145/3218603.3218617 pp. 52:1–52:6.

[27] J. Constantin, A. P. Burg, Z. Wang, A. Chattopadhyay, and G. Karakonstantis, "Statistical fault injection for impact-evaluation of timing errors on application performance," in *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*. ACM, 2016. doi: 10.1145/2897937.2898095 pp. 13:1–13:6.

[28] L. Mukhanov, D. S. Nikolopoulos, and B. R. de Supinski, "ALEA: fine-grain energy profiling with basic block sampling," in *2015 International Conference on Parallel Architecture and Compilation, PACT 2015, San Francisco, CA, USA, October 18-21, 2015*. IEEE Computer Society, 2015. doi: 10.1109/PACT.2015.16 pp. 87–98.

[29] OpenRISC Community, "OpenRISC 1000 architecture manual.".

[30] X. Jiao, Y. Jiang, A. Rahimi, and R. K. Gupta, "Slot: A supervised learning model to predict dynamic timing errors of functional units," in *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, D. Atienza and G. D. Natale, Eds. IEEE, 2017. doi: 10.23919/DATE.2017.7927168 pp. 1183–1188.

[31] NanGate FreePDK45 Open Cell Library, http://nangate.com.

[32] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan, "SALSA: systematic logic synthesis of approximate circuits," in *The 49th Annual Design Automation Conference 2012, DAC '12, San Francisco, CA, USA, June 3-7, 2012*, P. Groeneveld, D. Sciuto, and S. Hassoun, Eds. ACM, 2012. doi: 10.1145/2228360.2228504 pp. 796–801.

[33] Z. Wang, G. Karakonstantis, and A. Chattopadhyay, "A low overhead error confinement method based on application statistical characteristics," in *2016 Design, Automation & Test in Europe Conference & Exhibition, DATE 2016, Dresden, Germany, March 14-18, 2016*, L. Fanucci and J. Teich, Eds. IEEE, 2016, pp. 1168–1171.

[34] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *The 50th Annual Design Automation Conference 2013, DAC '13, Austin, TX, USA, May 29 - June 07, 2013*. ACM, 2013. doi: 10.1145/2463209.2488873 pp. 113:1–113:9.

[35] M. Gautschi, P. D. Schiavone, A. Traber, I. Loi, A. Pullini, D. Rossi, E. Flamand, F. K. Gürkaynak, and L. Benini, "Near-threshold RISC-V core with DSP extensions for scalable iot endpoint devices," *IEEE Trans. VLSI Syst.*, vol. 25, no. 10, pp. 2700–2713, 2017. doi: 10.1109/TVLSI.2017.2654506

[36] B. Moons and M. Verhelst, "Dvas: Dynamic voltage accuracy scaling for increased energy-efficiency in approximate computing," in *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, July 2015. doi: 10.1109/ISLPED.2015.7273520 pp. 237–242.

[37] J. Schlachter, V. Camus, K. V. Palem, and C. Enz, "Design and applications of approximate circuits by gate-level pruning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 5, pp. 1694–1702, May 2017. doi: 10.1109/TVLSI.2017.2657799

[38] H. Amrouch, B. Khaleghi, A. Gerstlauer, and J. Henkel, "Towards aging-induced approximations," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, June 2017. doi: 10.1145/3061639.3062331 pp. 1–6.

**Ioannis Tsiokanos** received the B.Sc. and M.Sc. degree from the Department of Electrical and Computer Engineering of University of Thessaly, Greece, in 2016. Currently pursues his Ph.D. on design of energy efficient and variation tolerant pipelined micro-architectures at the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK. His current research interests include low-power designs, fault tolerance circuits and hardware/software co-design with an emphasis on robustness

**Lev Mukhanov** is a Research Fellow with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast. He received the PH.D. degree in Computer Science from Moscow Engineering Physics Institute in 2009. His current research interests include compilers, energy efficient computing, fault tolerance, guardbanding, DRAM faults and DRAM reliability. He participated in various EU and UK projects, including EPSRC ALEA, CHIST-ERA DIVIDEND, EU Uniserver-H2020.

**Dimitrios S. Nikolopoulos** is the Director of the Institute of Electronics, Communications and Information Technology and a Professor in the School of Electronics, Electrical Engineering and Computer Science at Queen's University Belfast. His research explores system software for scalable computing systems. Dimitrios is the recipient of a Royal Society Wolfson Research Merit Award, an NSF CAREER Award, a DoE Early Career Principal Investigator Award, an IBMN Faulty Award and seven Best Paper awards from IEEE and ACM conferences. He holds a PhD ('00), MSc ('97) and BSc ('96) degrees in Computer Engineering and Informatics from the University of Patras.

**Georgios Karakonstantis** is an Assistant Professor at the School of Electronics, Electrical Engineering and Computer Science of Queen's University Belfast, United Kingdom and academic member of Queen's Global Research Institute of Electronics, Communications and Information Technology. He received the MSc and Ph.D. degree in Electrical and Computer Engineering from Purdue University, West-Lafayette, USA. In the past he worked at the Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland, and the Advanced Technology Group, Qualcomm Inc., San Diego, CA, USA. Since 2016, he directs the UniServer project funded by the Horizon 2020 research program of the European Commission. In 2012 he was awarded a Marie-Curie Fellowship by the European Commission and in 2010 won the Altera Innovate Design Contest. He serves as member of the program committee and referee of major conferences and journals and is a Stakeholder member of the HiPEAC network of excellence. He has published more than 75 papers in peer reviewed IEEE and ACM journals and conferences, and he is inventor of a US patent and author of two book chapters. His research focuses on energy-efficient and error-resilient computing and storage architectures for embedded and high-performance applications.